In the Claims

Claims 1, 13, 14, 37, 71 and 94 are amended.

Claims 62 and 63 were previously canceled.

Claims 1-61 and 64-96 remain in the application and are listed as follows:

1. (Currently Amended) A method for delivering software via a network comprising:

describing one or more software extensions using a hierarchical language, the extensions being configured for incorporation on a client, said describing defining one or more manifests containing at least one list of files comprising an extension; and

delivering the one or more manifests to the client via the network, the one or more manifests being configured for use in downloading the software extensions via the network, at least some of the extensions being downloadable by streaming extension files to the client in a manner that enables a user to begin to interact with the extension sooner than if the user had to wait for the entire extension to load, said manner being developed based on scenario runs in which files that are more likely to be first used by the user are downloaded before files that are less likely to be first used, and wherein files that are less likely to be used first can be downloaded via a background download process.

2. (Original) The method of claim 1, wherein the one or more manifests are configured to assist in organizing delivery of individual files listed in the one or more manifests.

3. (Original) The method of claim 1, wherein the one or more manifests are configured to assist in validating individual files listed in the one or more manifests.

4. (Original) The method of claim 1, wherein the one or more manifests are configured to assist in updating individual files listed in the one or more manifests.

5. (Original) The method of claim 1, wherein the one or more manifests describe individual file locations.

6. (Original) The method of claim 1, wherein the one or more manifests contain individual hashes for one or more of the listed files.

7. (Original) The method of claim 1, wherein the one or more manifests contain download directives for downloading the listed files.

8. (Original) The method of claim 1, wherein the one or more manifests are defined in a tag-based language.

9. (Original) The method of claim 1, wherein the one or more manifests are defined in extensible markup language (XML).

10. (Original) The method of claim 1, wherein the network comprises the Internet.

11. (Original) One or more computer-readable media comprising computer-readable instructions thereon which, when executed by a computer, cause the computer to implement the method of claim 1.

12. (Original) A computer programmed with instructions which, when executed by the computer, implemented the method of claim 1.

13. (Currently Amended) One or more computer-readable media comprising computer-readable instructions thereon which, when executed by a computer, cause the computer to:

describe one or more software extensions using extensible markup language (XML), the extensions being configured for incorporation on a client, said describing defining a manifest containing at least one list of files comprising an extension, the manifest being configured to assist in one or more of the following: organizing delivery of individual files listed in the manifest, validating individual files listed in the manifest, and updating individual files listed in the manifest; and

deliver the manifest to the client via the network, at least some of the extensions being downloadable by streaming extension files to the client in a manner that enables a user to begin to interact with the extension sooner than if the user had to wait for the entire extension to load, said manner being developed based on scenario runs in which files that are more likely to be first used by the user are downloaded before files that are less likely to be first used, and wherein

files that are less likely to be used first can be downloaded via a background download process.

14. (Currently Amended) A method for receiving software via a network comprising:

receiving a manifest that contains at least one list of files comprising a software extension that is to be downloaded via a network and incorporated on a client, the manifest being defined in extensible markup language (XML), the manifest being configured to assist in:

organizing delivery of the files,

validating individual files listed in the manifest, and

updating individual files listed in the manifest; and

downloading files from the list of files contained in the manifest;

wherein the extension is downloadable by streaming extension files to the client in a manner that enables a user to begin to interact with the extension sooner than if the user had to wait for the entire extension to load, said manner being developed based on scenario runs in which files that are more likely to be first used by the user are downloaded before files that are less likely to be first used, and wherein files that are less likely to be used first can be downloaded via a background download process.

15. (Original) The method of claim 14, wherein the software extension is to be incorporated into a software platform executing on the client.

16. (Original) The method of claim 14, wherein the downloading of the files takes place by downloading the files in an order that is described in the manifest.

17. (Original) The method of claim 14 further comprising validating one or more downloaded files using the manifest.

18. (Original) The method of claim 14 further comprising updating one or more files using the manifest.

19. (Original) One or more computer-readable media comprising computer-readable instructions thereon which, when executed by a computer, cause the computer to implement the method of claim 14.

20. (Original) A computer programmed with instructions which, when executed by the computer, implement the method of claim 14.

21. (Previously Presented) A data structure embodied on a computer-readable medium, the data structure comprising:

a list of one or more files that are utilized in a software extension that is configured to extend a software application executing on a client;

one or more hashes each of which being associated with a particular listed file; and

one or more file groups, individual files being associated with individual file groups, the file groups determining when particular files of the extension get downloaded to the client;

the data structure being configured to assist in delivering software extensions via the Internet.

22. (Original) The data structure of claim 21, wherein the file groups determine where files are stored on the client.

23. (Original) The data structure of claim 21, wherein the file groups determine how files are packaged for delivery.

24. (Original) The data structure of claim 21, wherein the file groups determine where files are stored on the client and how files are packaged for delivery.

25. (Original) The data structure of claim 21, wherein the file groups identify files that are to be downloaded before any other files.

26. (Original) The data structure of claim 21, wherein the file groups identify files that are to be downloaded for offline use.

27. (Original) The data structure of claim 21, wherein the file groups identify files that are only downloaded when they are required for the first time by a user.

28. (Original) The data structure of claim 21, wherein the file groups identify files that are downloaded on demand and provide content that is available only when a user is online.

29. (Original) The data structure of claim 21, wherein the file groups indicate file download priority.

30. (Original) The data structure of claim 21, wherein the one or more hashes are used for security.

31. (Original) The data structure of claim 21, wherein the one or more hashes are used for versioning.

32. (Original) The data structure of claim 21, further comprising a storage size that is associated with an amount of storage required by the extension.

33. (Original) The data structure of claim 21, further comprising one or more class identifiers for individual dynamic link libraries (DLLs).

34. (Original) The data structure of claim 21, further comprising one or more dynamic link library (DLL) load dependencies.

35. (Original) The data structure of claim 21, further comprising:

a storage size that is associated with an amount of storage required by the extension;

one or more class identifiers for individual dynamic link libraries (DLLs); and

one or more dynamic link library (DLL) load dependencies.

36. (Original) The data structure of claim 21 embodied as an extensible markup language (XML) file.

37. (Currently Amended) A method of providing software via a network comprising:

describing one or more software extensions using one or more extensible markup language (XML) files, the extensions being configured for incorporation in a software program executing on a client, individual XML files providing individual manifests that contain a list of files that comprise an extension; and

storing the XML files in a Web-accessible location;

wherein at least some of the extensions are downloadable by streaming extension files to the client in a manner that enables a user to begin to interact with the extension sooner than if the user had to wait for the entire extension to load, said manner being developed based on scenario runs in which files that are more likely to be first used by the user are downloaded before files that are less likely to be first used, and wherein files that are less likely to be used first can be downloaded via a background download process.

38. (Original) The method of claim 37 further comprising storing extension files associated with the XML files in a Web-accessible location.

39. (Original) The method of claim 37 further comprising:

storing extension files associated with the XML files in a Web-accessible location; and

providing one or more XML files and one or more associated extension files to a client via the network.

40. (Original) The method of claim 37, wherein individual manifests can contain one or more file hashes that can be used for file security.

41. (Original) The method of claim 37, wherein individual manifests can contain one or more file hashes that can be used for versioning.

42. (Original) The method of claim 37, wherein individual manifests comprise one or more file groups that determine when particular files are downloaded to the client.

43. (Original) The method of claim 42, wherein the file groups determine where files are stored on the client.

44. (Original) The method of claim 42, wherein the file groups determine how files are packaged.

45. (Original) The method of claim 42, wherein the file groups determine where files are stored on the client and how files are packaged.

46. (Original) One or more computer-readable media having computer-readable instructions thereon which, when executed by a computer, cause the computer to implement the method of claim 37.

47. (Original) A security method for downloading software extensions via the Internet comprising:

receiving, via the Internet, a package manifest containing a list of multiple files that comprise a software extension that is to be incorporated into an application program executing on a client, the list containing a hash for one or more of the files comprising the software extension;

receiving, via the Internet, the multiple files that are described in the package manifest;

creating a hash for one or more of the multiple received files; and

comparing the created hash of the one or more files with corresponding file hashes contained in the package manifest to ascertain whether one or more of the received file is secure.

48. (Original) The security method of claim 47, wherein the package manifest comprises an XML file.

49. (Original) One or more computer-readable media comprising computer-readable instructions thereon which, when executed by a computer, cause the computer to implement the method of claim 47.

50. (Original) One or more client computers programmed with instructions which, when executed by the one or more computers, cause the one or more computers to implement the method of claim 47.

51. (Original) An updating method for updating software extensions via the Internet comprising:

receiving, via the Internet, a package manifest containing a list of multiple files that comprise a newer version of a software extension that is to be incorporated into an application program executing on a client that contains an older software extension version, the list containing a hash for one or more of the files comprising the newer version of the software extension;

comparing one or more hashes that are received with one or more hashes of files from the older version of the software extension;

for any hashes of corresponding files from the different versions that are different, downloading a new file from a web server; and

for any hashes of corresponding files from the different versions that are the same, copying a file from an old local directory on the client to a new local directory on the client associated with the newer version of the extension.

52. (Original) The updating method of claim 51, wherein the package manifest comprises an XML file.

53. (Original) One or more computer-readable media comprising computer-readable instructions thereon which, when executed by a computer, cause the computer to implement the method of claim 51.

54. (Original) One or more client computers programmed with instructions which, when executed by the one or more computers, cause the one or more computers to implement the method of claim 51.

55. (Original) A data structure embodied on a computer-readable medium comprising:

one or more first tags indicative of associated file groups associated with an Internet-downloadable software extension that can extend an application program executing on a client; and

one or more second tags indicative of specific files that comprise the software extension.

56. (Original) The data structure of claim 55, wherein the one or more second tags can contain hashes for individual files.

57. (Original) The data structure of claim 55 further comprising one or more third tags indicative of file load dependencies.

58. (Original) The data structure of claim 55 further comprising one or more third tags indicative of COMClass IDs.

59. (Original) The data structure of claim 55 further comprising:

one or more third tags indicative of file load dependencies; and

one or more fourth tags indicative of COMClass IDs.

60. (Original) The data structure of claim 55, wherein the ordering of the file groups implicitly defines a download order of the files.

61. (Original) The data structure of claim 55, wherein the tags comprise XML tags.

62. (Cancelled).

63. (Cancelled).

64. (Original) A queue management method comprising:

defining a download queue that controls when files are to be downloaded to a client, the files pertaining to a software extension that is to be incorporated into an application program executing on the client;

ascertaining whether a user action at the client requires one or more files that are not currently being downloaded; and

manipulating the download queue responsive to a user action that requires one or more files that are not currently being downloaded so that the one or more required files are downloaded sooner than they would otherwise be.

65. (Original) The queue management method of claim 64, wherein the download queue contains multiple package objects each of which contains a list of one or more files that correspond to a software extension, and said manipulating comprises moving a package object to the head of the download queue.

66. (Original) The queue management method of claim 65, wherein said ascertaining comprises determining whether a required file is associated with a package object whose files are currently being downloaded.

67. (Original) The queue management method of claim 65, wherein said manipulating comprises:

ascertaining an identifier associated with a requested file; and

locating a package object with a corresponding identifier.

68. (Original) One or more computer-readable media comprising computer-readable instructions thereon which, when executed by a computer, cause the computer to implement the method of claim 64.

69. (Previously Presented) A method of creating software packages for delivery via the Internet comprising:

identifying end user features;

identifying shared dependencies between the end user features;

creating individual software packages for the end user features;

creating individual software packages for the shared dependencies; and

hosting the software packages on a web server;

wherein both of said acts of identifying and both of said acts of creating provide software extensions that are created in a uniform manner independent of end user input.

70. (Original) The method of claim 69, wherein the packages comprise one or more files that are associated with a single application program that provides multiple different functionalities, and the packages extend, in some way, the functionalities that are provided by the single application program.

71. (Currently Amended) An automated software tool comprising a package manifest creation tool configured to:

receive one or more input parameters pertaining to a package manifest that is to describe a software extension that is configured to extend a software application executing on a client; and

generate a package manifest that describes the extension, the package manifest being generated using a hierarchical language;

wherein the extension is downloadable by streaming extension files to the client in a manner that enables a user to begin to interact with the extension sooner than if the user had to wait for the entire extension to load, said manner being developed based on scenario runs in which files that are more likely to be first used by the user are downloaded before files that are less likely to be first used, and wherein files that are less likely to be used first can be downloaded via a background download process.

72. (Original) The automated software tool of claim 71, wherein the hierarchical language comprises a tag-based language.

73. (Original) The automated software tool of claim 71, wherein the hierarchical language comprises extensible markup language (XML).

74. (Original) The automated software tool of claim 71, wherein one input parameter specifies a directory containing files that are to be described in the package manifest.

75. (Original) The automated software tool of claim 71, wherein one input parameter comprises file group information and load dependencies.

76. (Original) The automated software tool of claim 71, wherein one input parameter comprises file usage statistics.

77. (Original) The automated software tool of claim 76, wherein the file usage statistics are ascertained from scenario runs.

78. (Original) The automated software tool of claim 77, wherein individual scenarios have individual priorities.

79. (Original) The automated software tool of claim 76, wherein the file usage statistics are ascertained from scenario runs that are collected by running logs on various scenarios.

80. (Original) The automated software tool of claim 76, wherein the file usage statistics are ascertained from scenario runs that are collected by running logs on various scenarios, the scenarios having individual checkpoints that separate one scenario from another.

81. (Previously Presented) The automated software tool of claim 76, wherein the file usage statistics are ascertained dynamically by building a knowledge base that describes tasks that users typically accomplish.

82. (Original) A method for creating a package manifest comprising:

receiving information pertaining to one or more extension directories that contain files that comprise a software extension that is to be incorporated into a software application program to extend the application program;

receiving, if any, information pertaining to file groups or load dependencies;

receiving, if any, information pertaining to file usage statistics; and

generating a package manifest based on the received information.

83. (Original) The method of claim 82, wherein the package manifest is generated in XML.

84. (Original) The method of claim 82, wherein the file usage statistics can be provided based upon scenario runs.

85. (Original) One or more computer-readable media having computer-readable instructions thereon which, when executed by a computer, cause the computer to implement the method of claim 82.

86. (Original) A method of providing software extensions via the Internet comprising:

assigning one or more files to one or more scenarios to provide multiple different scenarios that describe ways that a user interacts with a software application program;

assigning a priority to each of the scenarios;

sorting multiple files in accordance with their scenario priority or priorities; and

downloading sorted files in an order defined by said sorting.

87. (Original) The method of claim 86 further comprising sorting the multiple files in accordance with one or more file groups.

88. (Original) The method of claim 86 further comprising sorting the multiple files in accordance with a file usage order within one or more scenarios.

89. (Original) The method of claim 86 further comprising:

sorting the multiple files in accordance with one or more file groups; and

sorting the multiple files in accordance with a file usage order within one or more scenarios.

90. (Original) One or more computer-readable media having computer-readable instructions thereon which, when executed by a computer, cause the computer to implement the method of claim 86.

91. (Original) A method of ordering files for download to a client comprising:

sorting multiple files by one or more file groups;

sorting the multiple files based on scenario priority of one or more scenarios into which each file can be placed;

sorting the multiple files by file usage order within one or more scenarios.

92. (Original) The method of claim 91 further comprising determining a file group from a manifest defined in XML.

93. (Original) The method of claim 91 further comprising if no file group information is provided, assuming that all files of a particular type are of a higher priority than other files of different types.

94. (Currently Amended) The method of claim 91, wherein said sorting by file usage order comprises sorting the files according to the average order in which the files were downloaded within scenarios of their particular priority or priorities.

95. (Original) One or more computer-readable media having computer-readable instructions thereon which, when executed by a computer, cause the computer to implement the method of claim 91.

96. (Original) One or more server computers programmed with instructions which, when executed by the one or more server computers, cause the one or more server computers to implement the method of claim 91.